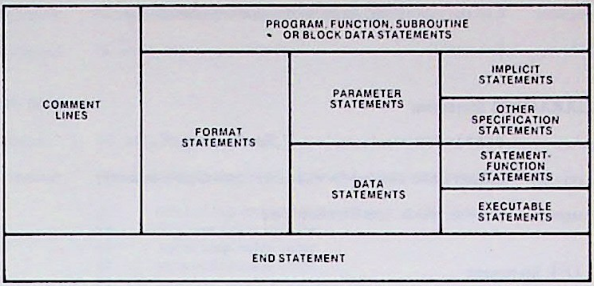
FORTRAN-86 POCKET REFERENCE

Order Number: 121571-001

CONTENTS

STATEMENT ORDER



FORTRAN STATEMENTS

ASSIGN Statement

Syntax: `ASSIGN` *stl* `TO` *name*

Function: Assign a statement label *stl* to an integer variable *name*

Category: Executable

Assignment Statement

Syntax: *name* = *exp*

Function: Assign the value of an expression *exp* to a variable *name*

Type: Arithmetic, Logical, Character

Category: Executable

BACKSPACE Statement

Syntax: `BACKSPACE` *unit*
`BACKSPACE` *arg-list*

Function: Position file connected to *unit* before preceding record where *unit* is the unit specifier and *arg-list* is

`[UNIT=]`*unit*	unit specifier
`IOSTAT=`*stname*	I/O status specifier
`ERR=`*stl*	error specifier

BACKSPACE is for sequential files only.

Category: Executable

BLOCK DATA Statement

Syntax: `BLOCK DATA[`*name*`]`

Function: Identify and optionally *name* a BLOCK DATA subprogram.

Category: Nonexecutable

CALL Statement

Syntax:	`CALL` *name*[(`[`*arg*[`,`*arg*]...])]
Function:	Call the subroutine, *name* with actual argument(s) *arg*.
Category:	Executable

CHARACTER Statement

Syntax:	`CHARACTER`[`*`*len*]*name*[`*`*len*][`,`*name*[`*`*len*]]...
Function:	Specify *name* and *len* for character type variable or array.
Category:	Nonexecutable, specification, type

CLOSE Statement

Syntax: `CLOSE(`*close-list*`)`

Function: Close the file described by *close-list*, where *close-list* is

[`UNIT=`]*unit*	unit specifier
`IOSTAT=`*stname*	I/O status specifier
`ERR=`*stl*	error specifier
`STATUS=`*stat*	file disposition specifier

Category: Executable

Comment Line

Syntax:	The character 'C' or asterisk (*) in position 1; any other characters in positions 2-72.
Function:	Program documentation
Category:	Nonexecutable

COMMON Statement

Syntax:	`COMMON`[`/`*name*]`/`]*nlist*[[`,`]`/`*name*`/`*nlist*]...
Function:	Name and define the contents of COMMON block(s), *name*. If *name* is not specified, a blank COMMON is defined.
Category:	Nonexecutable, specification

CONTINUE Statement

Syntax:	`CONTINUE`
Function:	No effect unless this is the terminal statement of a DO loop; then action depends on the DO variable.

DATA Statement

Syntax:	`DATA` *nlist*`/`*clist*...
Function:	Assign values in *clist* to the items in *nlist*.
Category:	Nonexecutable

DIMENSION Statement

Syntax:	`DIMENSION` *array*`(`*d*`)[,`*array*`(`*d*`)]...`
Function:	Name *array*(s) and define dimension(s) *d*.
Category:	Nonexecutable, specification

DO Statement

Syntax:	`DO` *stl*`[,]`*var*`=`*e1*`,`*e2*`[,`*e3*`]`
Function:	Define the beginning of DO loop and set up loop counters where

stl	label of last (executable) statement in DO loop
var	DO loop index variable
e1	initial loop index value
e2	loop termination value
e3	loop increment/decrement value

Category: Executable

DOUBLE PRECISION Statement

Syntax:	`DOUBLE PRECISION` *name*`[,`*name*`]...`
Function:	Specify name(s) for a double precision type variable or array.
Category:	Nonexecutable, specification, type

ELSE Statement

Syntax:	`ELSE`
Function:	Provides alternate execution path from IF or ELSE IF.
Category:	Executable, block IF

ELSE IF Statement

Syntax:	`ELSE IF(`*exp*`)THEN`
Function:	Continue execution if expression *exp* is TRUE.
Category:	Executable, Block IF

END Statement

Syntax:	`END`
Function:	Terminate main program; return from subprogram; mark end of program unit.
Category:	Executable

END IF Statement

Syntax:	`END IF`
Function:	Mark end of IF block; continue execution.
Category:	Executable, block IF

ENDFILE Statement

Syntax: `ENDFILE` *unit*
`ENDFILE(`*arg-list*`)`

Function: Write end-of-file record on file connected to *unit* where *unit* is the unit specifier and *arg-list* is

[`UNIT=`]*unit*	unit specifier
`IOSTAT=`*stname*	I/O status specifier
`ERR=`*stl*	error specifier

ENDFILE is for sequential files only.

Category: Executable

EQUIVALENCE Statement

Syntax: `EQUIVALENCE(`*nlist*`)`[`,(`*nlist*`)`]...

Function: Allow entries in *nlist* to share the same storage area.

Category: Nonexecutable, specification

EXTERNAL Statement

Syntax: `EXTERNAL` *name*[`,`*name*]...

Function: Allows the name of an external/dummy procedure name to be used as an actual argument.

Category: Nonexecutable, specification

FORMAT Statement

Syntax: *stl* `FORMAT(`[*flist*]`)`

Function: Specifies the format of formatted I/O data where *flist* includes the following repeatable and nonrepeatable edit descriptors

Repeatable		Nonrepeatable	
Iw	integer	'string'	literal
Fw.d	real	nHstring	Hollerith
Ew.d[Ee]	real	nX	record position
Dw.d	real	/	record termination
Gw.d[Ee]	real	kP	scale factor
Lw	logical	BN	blank
A[w]	alphanumeric	BZ	blank
Bw	binary	$	alternate-record termination
Zw	hexadecimal		

Category: Nonexecutable

FUNCTION Statement

Syntax: [*type*]`FUNCTION` *name*`(`[*arg*[`,`*arg*]...]`)`

Function: Name the FUNCTION subprogram and define its type and dummy argument(s).

Category: Nonexecutable

GO TO Statements

Syntax: `GO TO` *stl*
`GO TO (`*stl*`[,`*stl*`]...)`*exp*
`GO TO` *name*`[(`*stl*`[,`*stl*`]...)]`

Function: Transfer control to statement labelled *stl* or ASSIGNED to variable *name*. The first branches unconditionally, the second branches based on the value of the integer expression *exp*; the third branches unconditionally, but statement label corresponding to *name* must be included in list.

Category: Executable

IF Statements

Syntax: `IF(`*exp*`)`*s1*`,`*s2*`,`*s3*
`IF(`*exp*`)`*stmt*
`IF(`*exp*`)THEN`

Function: Transfer control to a specified statement or perform specified action(s) based on the value of the expression *exp*. In the first format, *exp* is an arithmetic expression and *s1*, *s2*, and *s3* are statement labels; control passes to:

s1 if exp<0
s2 if exp=0
s3 if exp>0

In the second format, the statement *stmt* is executed if the logical expression is TRUE. Third format introduces IF block; statements following IF-THEN are executed if logical expression is TRUE.

Category: Executable

IMPLICIT Statement

Syntax: `IMPLICIT` *ntype*`(`*let*`[`*let*`]...)...`

Function: Define implicit typing for variable names whose first letter is *let* or in the range *let-let*.

Category: Nonexecutable, specification

INTEGER Statement

Syntax: `INTEGER[*`*len*`]`*name*`[*`*len*`][`*name*`[*`*len*`]]...`

Function: Define *name* to be of type integer with length *len*.

Category: Nonexecutable, specification, type

INTRINSIC Statement

Syntax: `INTRINSIC` *name*`[,`*name*`]...`

Function: Allow intrinsic function(s) to be used as actual argument(s).

Category: Nonexecutable, specification

LOGICAL Statement

Syntax:	`LOGICAL[*`*len*`]`*name*`[*`*len*`][,`*name*`[*`*len*`]]...`
Function:	Define *name* to be of type logical with length *len*
Category:	Nonexecutable, specification, type

OPEN Statement

Syntax: `OPEN(`*open-list*`)`

Function: Open the specified file with *open-list* consisting of the following:

`[UNIT=]`*unit*	unit specifier
`IOSTAT=`*stname*	I/O status specifier
`ERR=`*stl*	error specifier
`FILE=`*fname*	filename specifier
`STATUS=`*stat*	file status specifier
`ACCESS=`*acc*	access method specifier
`FORM=`*fmat*	formatting specifier
`RECL=`*reclen*	record length specifier
`BLANK=`*blnk*	blank specifier
`CARRIAGE=`*car*	carriage control specifier

Category: Executable

PAUSE Statement

Syntax:	`PAUSE[`*msg*`]`
Function:	Halt program execution; resume under control of external signal; *msg* is 1-5 digits or a character constant.
Category:	Executable

PARAMETER Statement

Syntax:	`PARAMETER(`*name*`=`*exp*`...)`
Function:	Assigns a *name* to a constant expression *exp*.
Category:	Nonexecutable, specification

PRINT Statement

Syntax:	`PRINT` *f*`[,`*outlist*`]`
Function:	Output items in *outlist* to preconnected unit in format specified by *f*.
Category:	Executable

PROGRAM Statement

Syntax:	`PROGRAM` *name*
Function:	Optionally name main-program unit. If missing, the compiler will assign @MAIN as the program name.
Category:	Nonexecutable

READ Statement

Syntax: `READ`(*ctl-list*)[*inlist*]
`READ` *f*[,*inlist*]

Function: Input items in *inlist* as directed by specified controls in *ctl-list*

[UNIT=]*unit*	unit specifier
[FMT=]*f*	format specifier
REC=*recno*	record number specifier
IOSTAT=*stname*	I/O status specifier
ERR=*stl*	error specifier
END=*stl*	end-of-file specifier

Second format is for preconnected units; *f* is the format specifier.

Category: Executable

REAL Statement

Syntax: `REAL`[**len*]*name*[**len*][,*name*[**len*]]...

Function: Define *name* to be of type real with length *len*.

Category: Nonexecutable, specification, type

RETURN Statement

Syntax: `RETURN`

Function: Return from FUNCTION or SUBROUTINE subprogram.

Category: Executable

REWIND Statement

Syntax: `REWIND` *unit*
`REWIND`(*arg-list*)

Function: Reposition file connected to *unit* at its initial point with *arg-list* including:

[UNIT=]*unit*	unit specifier
IOSTAT=*stname*	I/O status specifier
ERR=*stl*	error specifier

REWIND is for sequential files only.

Category: Executable

SAVE Statement

Syntax: `SAVE`/*name*/[,/*name*/]...

Function: Save data in common block *name* on return from subprogram.

Category: Nonexecutable, specification

Statement Function Statement

Syntax: *name*([*arg*[,*arg*]...])=*exp*

Function: Define function *name*

Category: Nonexecutable

STOP Statement

Syntax: STOP[*msg*]

Function: Terminate program execution, with optional message, *msg*.

Category: Executable

SUBROUTINE Statement

Syntax: SUBROUTINE *name*[([*arg*[,*arg*]...])]

Function: Define SUBROUTINE subprogram *name* with dummy argument(s) *arg*.

Category: Nonexecutable

TEMPREAL Statement

Syntax: TEMPREAL *name*[,*name*]...

Function: Define *name* to be of type tempreal.

Category: Nonexecutable, specification, type

WRITE Statement

Syntax: WRITE(*ctl-list*) [*outlist*]

Function: Output items in *outlist* as directed by controls in *ctl-list* including

[UNIT=]*unit*	unit specifier
[FMT=]*f*	format specifier
REC=*recno*	record number specifier
IOSTAT=*stname*	I/O status specifier
ERR=*stl*	error specifier

Intrinsic Functions

Type-Conversion Functions

Generic Name	Specific Name	Category	Function	Type	
				Arguments	Result
INT		Type Conversion	Convert to INTEGER	INTEGER	INTEGER
				INTEGER*1	INTEGER
				INTEGER*2	INTEGER
				INTEGER*4	INTEGER
	INT			REAL*4	INTEGER
	IFIX			REAL*4	INTEGER
				REAL*8	INTEGER
	IDINT			DOUBLE PRECISION	INTEGER
				TEMPREAL	INTEGER
INT1		Type Conversion	Convert to INTEGER*1	INTEGER	INTEGER*1
				INTEGER*1	INTEGER*1
				INTEGER*2	INTEGER*1
				INTEGER*4	INTEGER*1
				REAL*4	INTEGER*1
				REAL*8	INTEGER*1
				DOUBLE PRECISION	INTEGER*1
				TEMPREAL	INTEGER*1

Generic Name	Specific Name	Category	Function	Type	
				Arguments	Result
INT2		Type Conversion	Convert to INTEGER*2	INTEGER	INTEGER*2
				INTEGER*1	INTEGER*2
				INTEGER*2	INTEGER*2
				INTEGER*4	INTEGER*2
				REAL*4	INTEGER*2
				REAL*8	INTEGER*2
				DOUBLE PRECISION	INTEGER*2
				TEMPREAL	INTEGER*2
INT4		Type Conversion	Convert to INTEGER*4	INTEGER	INTEGER*4
				INTEGER*1	INTEGER*4
				INTEGER*2	INTEGER*4
				INTEGER*4	INTEGER*4
				REAL*4	INTEGER*4
				REAL*8	INTEGER*4
				DOUBLE PRECISION	INTEGER*4
				TEMPREAL	INTEGER*4
REAL		Type Conversion	Convert to REAL	INTEGER	REAL*4
	FLOAT			INTEGER	REAL*4
				INTEGER*1	REAL*4
	FLOAT			INTEGER*1	REAL*4
				INTEGER*2	REAL*4
	FLOAT			INTEGER*2	REAL*4
				INTEGER*4	REAL*4
	FLOAT			INTEGER*4	REAL*4
				REAL*4	REAL*4
				REAL*8	REAL*4
	SNGL			DOUBLE PRECISION	REAL*4
				TEMPREAL	REAL*4
DBLE		Type Conversion	Convert to DOUBLE PRECISION	INTEGER	DOUBLE PRECISION
				INTEGER*1	DOUBLE PRECISION
				INTEGER*2	DOUBLE PRECISION
				INTEGER*4	DOUBLE PRECISION
				REAL*4	DOUBLE PRECISION
				REAL*8	DOUBLE PRECISION
				DOUBLE PRECISION	DOUBLE PRECISION
				TEMPREAL	DOUBLE PRECISION
TREAL		Type Conversion	Convert to TEMPREAL	INTEGER	TEMPREAL
				INTEGER*1	TEMPREAL
				INTEGER*2	TEMPREAL
				INTEGER*4	TEMPREAL
				REAL*4	TEMPREAL
				REAL*8	TEMPREAL
				DOUBLE PRECISION	TEMPREAL
				TEMPREAL	TEMPREAL
	ICHAR	Type Conversion	Convert CHAR to INTEGER	CHARACTER	INTEGER
CHAR		Type Conversion	Convert INTEGER to CHARACTER	INTEGER	CHARACTER
				INTEGER*1	CHARACTER
				INTEGER*2	CHARACTER
				INTEGER*4	CHARACTER

Truncation and Rounding Functions

Generic Name	Specific Name	Category	Function	Type	
				Arguments	Results
AINT		Truncation	Truncate Argument	REAL*4	REAL*4
	DINT			REAL*8	REAL*8
	DINT			DOUBLE PRECISION	DOUBLE PRECISION
				TEMPREAL	TEMPREAL
ANINT		Rounding	Round to Nearest Whole Number	REAL*4	REAL*4
	DNINT			REAL*8	REAL*8
	DNINT			DOUBLE PRECISION	DOUBLE PRECISION
				TEMPREAL	TEMPREAL
NINT		Rounding	Round to Integer	REAL*4	INTEGER
	IDNINT			REAL*8	INTEGER
	IDNINT			DOUBLE PRECISION	INTEGER
				TEMPREAL	INTEGER
RINT		Rounding	Round to Even Whole Number	REAL*4	REAL*4
	DRINT			REAL*8	REAL*8
	DRINT			DOUBLE PRECISION	DOUBLE PRECISION
				TEMPREAL	TEMPREAL
IRINT		Rounding	Round to Even Integer	REAL*4	INTEGER
	IDRINT			REAL*8	INTEGER
	IDRINT			DOUBLE PRECISION	INTEGER
				TEMPREAL	INTEGER

Remainder Functions

Generic Name	Specific Name	Category	Function	Type	
				Arguments	Results
MOD		Remainder	arg1-AINT (arg1/arg2) *arg2	INTEGER	INTEGER
				INTEGER*1	INTEGER*1
				INTEGER*2	INTEGER*2
				INTEGER*4	INTEGER*4
	AMOD			REAL*4	REAL*4
	DMOD			REAL*8	REAL*8
	DMOD			DOUBLE PRECISION	DOUBLE PRECISION
				TEMPREAL	TEMPREAL
RMD	IRMD	Remainder	arg1-RINT (arg1/arg2) *arg2	INTEGER	INTEGER
				INTEGER*1	INTEGER*1
				INTEGER*2	INTEGER*2
				INTEGER*4	INTEGER*4
				REAL*4	REAL*4
	DRMD			REAL*8	REAL*8
	DRMD			DOUBLE PRECISION	DOUBLE PRECISION
				TEMPREAL	TEMPREAL

Absolute Value, Sign Transfer, Positive Difference, and Double Precision Product Functions

Generic Name	Specific Name	Category	Function	Type	
				Arguments	Results
ABS	IABS	Absolute Value	Return Absolute Value	INTEGER	INTEGER
				INTEGER*1	INTEGER*1
				INTEGER*2	INTEGER*2
				INTEGER*4	INTEGER*4
				REAL*4	REAL*4
	DABS			REAL*8	REAL*8
	DABS			DOUBLE PRECISION	DOUBLE PRECISION
				TEMPREAL	TEMPREAL
SIGN	ISIGN	Sign Transfer	Transfer Sign of arg2 to arg1 sign(y,x)= 1y1,x≥0 −1y1,x≤0	INTEGER	INTEGER
				INTEGER*1	INTEGER*1
				INTEGER*2	INTEGER*2
				INTEGER*4	INTEGER*4
				REAL*4	REAL*4
	DSIGN			REAL*8	REAL*8
	DSIGN			DOUBLE PRECISION	DOUBLE PRECISION
				TEMPREAL	TEMPREAL
DIM	IDIM	Positive Difference	Return arg1−arg2 if arg1>arg2 else 0	INTEGER	INTEGER
				INTEGER*1	INTEGER*1
				INTEGER*2	INTEGER*2
				INTEGER*4	INTEGER*4
				REAL*4	REAL*4
				REAL*8	REAL*8
	DDIM			DOUBLE PRECISION	DOUBLE PRECISION
	DDIM			TEMPREAL	TEMPREAL
DPROD		Double Precision Product	Multiply arg1 by arg2	REAL*4	DOUBLE PRECISION

Choosing the Largest or Smallest Value Functions

Generic Name	Specific Name	Category	Function	Type	
				Arguments	Results
MAX	MAX0	Largest Value	Choose Largest Value in List	INTEGER	INTEGER
				INTEGER*1	INTEGER*1
				INTEGER*2	INTEGER*2
				INTEGER*4	INTEGER*4
	AMAX1			REAL*4	REAL*4
				REAL*8	REAL*8
	DMAX1			DOUBLE PRECISION	DOUBLE PRECISION
				TEMPREAL	TEMPREAL
AMAX0		Largest Value	Choose Largest Value in List	INTEGER	REAL*4
				INTEGER*1	REAL*4
				INTEGER*2	REAL*4
				INTEGER*4	REAL*4
	MAX1			REAL*4	INTEGER
MIN	MIN0	Smallest Value	Choose Smallest Value in List	INTEGER	INTEGER
				INTEGER*1	INTEGER*1
				INTEGER*2	INTEGER*2
				INTEGER*4	INTEGER*4
	AMIN1			REAL*4	REAL*4
				REAL*8	REAL*8
	DMIN1			DOUBLE PRECISION	DOUBLE PRECISION
				TEMPREAL	TEMPREAL
AMIN0		Smallest Value	Choose Smallest Value in List	INTEGER	REAL*4
				INTEGER*1	REAL*4
				INTEGER*2	REAL*4
				INTEGER*4	REAL*4
	MIN1			REAL*4	INTEGER

Length and Index Functions

Generic Name	Specific Name	Category	Function	Type	
				Arguments	Result
	LEN	Length	Determine the Length of Character Entity	CHARACTER	INTEGER
	INDEX	Index of Substring	Return Location of Substring arg2 in String arg1	CHARACTER	INTEGER

Arithmetic Functions

Generic Name	Specific Name	Category	Function	Type	
				Arguments	Results
SQRT	DQRT DSQRT DSQRT	Arithmetic	Return Square Root	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL
EXP	 DEXP DEXP	Arithmetic	Return e Raised to Power of Argument	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL
LOG	ALOG DLOG DLOG	Arithmetic	Return Natural Logarithm	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL
LOG10	ALOG10 DLOG10 DLOG10	Arithmetic	Return Common Logarithm	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL

Trigonometric Functions

Generic Name	Specific Name	Category	Function	Type	
				Arguments	Results
SIN	 DSIN DSIN	Trigonometric	Return Sine	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL
COS	 DCOS DCOS	Trigonometric	Return Cosine	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL
TAN	 DTAN DTAN	Trigonometric	Return Tangent	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL
ASIN	 DASIN DASIN	Trigonometric	Return Arcsine	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL

Trigonometric Functions (Cont'd.)

Generic Name	Specific Name	Category	Function	Type	
				Arguments	Results
ACOS	DACOS DACOS	Trigonometric	Return Arccosine	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL
ATAN	DATAN DATAN	Trigonometric	Return Arctangent with one Argument	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL
ATAN2	DATAN2 DATAN2	Trigonometric	Return Arctangent with two Arguments	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL

Hyperbolic Functions

Generic Name	Specific Name	Category	Function	Type	
				Arguments	Results
SINH	DSINH DSINH	Hyperbolic	Return Hyperbolic Sine	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL
COSH	DCOSH DCOSH	Hyperbolic	Return Hyperbolic Cosine	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL
TANH	DTANH DTANH	Hyperbolic	Return Hyperboic Tangent	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL	REAL*4 REAL*8 DOUBLE PRECISION TEMPREAL

Lexical Relationship Functions

Generic Name	Specific Name	Category	Function	Type	
				Arguments	Results
LGE	LGE	Lexical Relationship	Lexically Greater or Equal	CHARACTER	LOGICAL
	LGT	Lexical Relationship	Lexically Greater	CHARACTER	LOGICAL
	LLE	Lexical Relationship	Lexically Less or Equal	CHARACTER	LOGICAL
	LLT	Lexical Relationship	Lexically Less	CHARACTER	LOGICAL

8087 Control Intrinsics

Form	Function	8087 Instruction Generated
STSW87	Store 87 Status Word	PUSHF CLI FNSTSW @ wd FNCLEX FWAIT POPF
LDCW87	Load 87 Control Word	PUSHF CLI FNLDCW @ wd POPF
STCW87(wd)	Store 87 Control Word	PUSHF CLI FNSTCW @ wd POPF
SAV87(st)	Save 87 State	PUSHF CLI FNSAVE @ st FWAIT POPF
RST87	Restore 87 State	FRSTOR @ st FWAIT

Where: wd = any INTEGER*2 variable
st = any array of at least 94 bytes

Intrinsic Subroutines

CALL INPUT(*port*,*var*)
CALL OUTPUT(*port*,*var*)
CALL INW(*port*,*var*)
CALL OUTW(*port*,*var*)

Statement Functions

name ([*arg* , [*arg* , ...]]) =*exp*

Compiler Controls

Types of Controls

Category	Primary Controls	General Controls
Listing Content	PRINT SYMBOLS XREF	LIST CODE
Listing Format	TITLE PAGEWIDTH PAGELENGTH	SUBTITLE EJECT
Input Format	DO66/DO77 STORAGE	INCLUDE FREEFORM
Object File	OBJECT ERRORLIMIT DEBUG	INTERRUPT REENTRANT
Control Status	IGNORE	

Controls and Their Abbreviations

Control	Abbreviation
CODE	CO
DEBUG	DB
+ DO66/DO77	none
+ EJECT	EJ
ERRORLIMIT	EL
FREEFORM	FF
+ IGNORE	IN
+ INCLUDE	IC
+ INTERRUPT	IT
LIST	LI
OBJECT	OJ
+ PAGELENGTH	PL
+ PAGEWIDTH	PW
PRINT	PR
+ REENTRANT	RE
+ STORAGE	SR
+ SUBTITLE	ST
SYMBOLS	SB
+ TITLE	TT
XREF	XR

Compiler Invocation

[:F*n*:]RUN[:F*n*:]FORT86[:F*n*:]*source*[*controls*]

Run-Time Support Libraries

- F86RN0.LIB, F86RN1.LIB, F86RN2.LIB, F86RN3.LIB, F86RN4.LIB, and RTNULL.LIB—run-time support libraries.
- CEL.LIB—floating-point intrinsic function library.
- 87ERH.LIB—floating-point error handler.
- 8087.LIB—8087 Numeric Data Processor interface library.
- E8087, and E8087.LIB—8087 Emulator and interface library.
- 87NULL.LIB—support library that resolves references if no 8087 processor is used.

LINK86 Invocation

RUN[:F*n*:]LINK86 *input-list*[TO *object-file*][*controls*]

LOC86 Invocation

RUN[:F*n*:]LOC86 *input-file*[TO *object-file*][*controls*]

ASCII CHARACTER	HEX	FORTRAN-86 CHARACTER	ASCII CHARACTER	HEX	FORTRAN-86 CHARACTER
NUL	00	no	@	40	no
SOH	01	no	A	41	yes
STX	02	no	B	42	yes
ETX	03	no	C	43	yes
EOT	04	no	D	44	yes
ENQ	05	no	E	45	yes
ACK	06	no	F	46	yes
BEL	07	no	G	47	yes
BS	08	no	H	48	yes
HT	09	no	I	49	yes
LF	0A	no	J	4A	yes
VT	0B	no	K	4B	yes
FF	0C	no	L	4C	yes
CR	0D	no	M	4D	yes
SO	0E	no	N	4E	yes
SI	0F	no	O	4F	yes
DLE	10	no	P	50	yes
DC1	11	no	Q	51	yes
DC2	12	no	R	52	yes
DC3	13	no	S	53	yes
DC4	14	no	T	54	yes
NAK	15	no	U	55	yes
SYN	16	no	V	56	yes
ETB	17	no	W	57	yes
CAN	18	no	X	58	yes
EM	19	no	Y	59	yes
SUB	1A	no	Z	5A	yes
ESC	1B	no	[	5B	no
FS	1C	no	\	5C	no
GS	1D	no	]	5D	no
RS	1E	no	^ (↑)	5E	no
US	1F	no	_	5F	no
space	20	yes	`	60	no
!	21	no	a	61	yes
"	22	no	b	62	yes
#	23	yes	c	63	yes
$	24	yes	d	64	yes
%	25	no	e	65	yes
&	26	no	f	66	yes
'	27	yes	g	67	yes
(	28	yes	h	68	yes
)	29	yes	i	69	yes
*	2A	yes	j	6A	yes
+	2B	yes	k	6B	yes
,	2C	yes	l	6C	yes
-	2D	yes	m	6D	yes
.	2E	yes	n	6E	yes
/	2F	yes	o	6F	yes
0	30	yes	p	70	yes
1	31	yes	q	71	yes
2	32	yes	r	72	yes
3	33	yes	s	73	yes
4	34	yes	t	74	yes
5	35	yes	u	75	yes
6	36	yes	v	76	yes
7	37	yes	w	77	yes
8	38	yes	x	78	yes
9	39	yes	y	79	yes
:	3A	no	z	7A	yes
;	3B	no	{	7B	no
<	3C	no	|	7C	no
=	3D	yes	}	7D	no
>	3E	no	~	7E	no
?	3F	no	DEL	7F	no

3065 Bowers Avenue, Santa Clara, California 95051
(408) 987-8080

Printed in U.S.A.